CISC 324, Winter 2015 -- Assignment 5
This assignment is due Tuesday, March 24, in lecture
Lab 6 is due Friday, March 27, electronic submission by midnight

Readings

- **More memory management** Continue the readings from assignment 4: text Ch. 8-9 and course reader pp. 49–59.
- **Monitors** Skim textbook Section 5.8 (Section 6.7 in 8th edition) and course reader page 40-41. Details about Java monitors are provided with lab 5 instructions, on pages 82-84 of the course reader.
- **Message passing** Course reader pages 41–42, Textbook Sec. 3.4.2. [See section 3.5 if you are interested in examples of message passing systems such as POSIX, Mach and Windows. We do not have time to cover these systems in lecture, so they won’t be on exams.]
- **Introduction to CPU scheduling** Textbook Sec. 6.1-6.2. This assignment only covers the definition of **turnaround** and **throughput**. The CPU scheduling algorithms in Section 6.3 are covered in Assignment 6.

**Memory Management – Questions 1 to 5**

1) A paging system uses 32 bit virtual addresses. Each page table entry occupies 4 bytes. [To get ideas on how to answer parts (a) and (b) of this question, refer to the “Paging with a Large Virtual Memory” example on pages 58-59 of the course reader. Also see the discussion of hierarchical paging in textbook section 8.6.1.]

(a) What is the smallest page size that allows the page table to fit into one page?

Hint: Since the page size is always a power of two, the answer has the form “2^k bytes”, where you have to find the value of k. Do this by writing down an expression that you can solve for k. If you have trouble finding the right expression to use, then you can use the following approach -- this is a generalized binary search that I have found useful in many problem-solving situations. Start by guessing some value for k, and check whether that value is too large or too small. For example, guess k=8 so the lowest 8 bits of the virtual address are the page offset. This leaves the top 24 bits to act as the page number, so virtual memory is divided into 2^{24} pages. As a result, the page table contains 2^{24} entries of 4 bytes = 2^{26} bytes; this page table is much larger than the page size of 2^8 bytes. Thus we conclude that k needs to be larger than 8. Using a next guess of k=20, you will find that k should be smaller than 20. At this point, you can guess a value of k in the middle of the range 8..20 (analogous to binary search), or you can use these examples to formulate an expression that can be solved for k.

(b) A page size of 2^{10} bytes is chosen. This means that the page table is hierarchical, with multiple levels. State how the 32 bit virtual address is divided into fields: How many fields are there, and how many bits are in each field?

2) Consider the following sequence of memory references from a program. Each memory reference is a 12 bit number, expressed as three hexadecimal digits.

   019, 01A, 1E4, 170, 073, 30E, 185, 24B, 24C, 430, 458, 364

(a) What is the reference string, assuming a page size of 100_{10} bytes? (Reference strings are defined at the end of section 9.4.1 of the textbook.)

(b) Find the page fault rate for the reference string in part (a): assume that 2 frames of main memory are available to the program and the FIFO page replacement algorithm is used. Note that the page fault rate is calculated as “number of page faults” divided by “number of memory references used to form the reference string”.

(c) Repeat (b) using the LRU replacement algorithm.

(d) Repeat (b) using the optimal replacement algorithm.

3) Assume you have a reference string for a process with m frames, and initially all m frames are empty. The reference string has length p, with n distinct page numbers occurring in it. Give an upper bound and a lower bound on the number of page faults. Your bounds should hold for any page-replacement algorithm. Briefly justify your answers.
4) This program swaps the elements in the first half of array \( A \) with the elements in the second half of the array:

```plaintext
var A: array[1..1000] of integer; // A is an array indexed from 1 to 1000
  temp1, temp2, i : integer;
  // Code to initialize A has been omitted. Here is the loop to swap elements in A:
  for (i := 1 to 500) do {   // i is an index for the first half of the array.
    temp1 := A[i];       // This code swaps A[i] and A[500+i].
    temp2 := A[500+i];
    A[i] := temp2;
    A[500+i] := temp1;
  }
```
Assume that 100 integers fit into a page, so array \( A \) fits into 10 pages. Initially, all of \( A \) is on disk. How many page faults occur during program execution, in cases (a) and (b)? Use Least Recently Used (LRU) page replacement. [Note that in this problem we only consider page faults that result from accessing \( A \): assume there are no page faults from fetching instructions.]

(a) Five page frames of size 100 integers are allocated to array \( A \)
(b) One page frame of size 100 integers is allocated to array \( A \)

5) Briefly describe what thrashing is. How can the operating system detect when thrashing occurs? What should the operating system do when thrashing occurs? [Refer to textbook Section 9.6.1 for a discussion of thrashing.]

**Monitors** (no assignment question)

I do not emphasize monitors in this course because Java provides such a limited form of monitors, with only one condition variable per monitor. Lab 5/6 gives you experience with Java monitors, expressed using the keyword `synchronized` and the methods `wait()`, `notify()` and `notifyAll()`.

Final exam questions about writing or analyzing concurrent code will use semaphores, not monitors. The final exam might contain some short-answer questions about monitors, to test that you know what a monitor is. From your lab 5/6 experience, you should know that a monitor is a module of source code containing declaration, initialization and access procedures for shared data. The monitor automatically provides mutually-exclusive access, meaning that only one process at a time can be executing an access procedure.

**Message Passing**

6) (a) State two reasons why shared memory isn't available in all computer systems.

(b) I start up one process on a computer in Kingston and another process on a computer in Winnipeg, and these two computers are connected by a network. Can I use semaphores to synchronize these processes?

(c) Does a remote procedure call use synchronous or asynchronous message passing?

**CPU Scheduling**

7) What is the difference between **turnaround time** and **throughput**? Describe one scenario in which throughput is improved without changing average turnaround time. Describe another scenario in which turnaround time is improved without changing throughput. The scenarios could include changes to the number of jobs presented to a computer system, changes to the number of CPUs in the system, and changes to the speed of the CPUs in the system.

8) Compute the average turnaround time and the throughput for each of the following four situations. All jobs are sharing the use of a single I/O device; jobs have to wait if the I/O device is in use.

a) Situation A: One job is submitted every minute. Each job takes 30 seconds of I/O time, and CPU use is negligible.

b) Situation B: Same as situation A, but a job is submitted once every 30 seconds.

c) Situation C: Same as situation A, but a job is submitted once every 15 seconds.

d) Situation D: Same as situation A, but we've substituted an I/O device that is twice as fast.